Report No. ICST/HLNP - 80-1

NATIONAL BUREAU OF STANDARDS
Institute for Computer Sciences and Technology
Systems and Network Architecture Division

D R A F T   R E P O R T

FEATURES OF THE TRANSPORT AND SESSION PROTOCOLS

March 1980

Prepared by:
Bolt Beranek and Newman Inc.

Under Contract No. NB79SBCA0092

FOREWORD BY THE NATIONAL BUREAU OF STANDARDS

The National Bureau of Standards, Institute for Computer Sciences and Technology (ICST), has initiated a program to develop computer network protocol standards as Federal Information Processing Standards (FIPS). The objectives of this program are (1) to make possible distributed computer networks, and (2) to enable the interconnection of different components selected based on cost, performance, availability, and competitive procurements.

This draft report is one of a series of draft reports being prepared under the network protocol standards program for distribution to government agencies, voluntary standards organizations, computer and communications equipment manufacturers, and other interested parties. We urge readers to provide comments and to further interact with us on this draft report. Written comments should address both the advantages and disadvantages (from the reader's viewpoint) of individual features described in this draft. Responses should be directed to the address below, and NOT to the NBS Contractor that prepared the draft report.

Reply to:

National Bureau of Standards (Code HLNP-80-1)
Institute for Computer Sciences and Technology
Systems and Network Architecture Division
Technology Building Room B218
Washington, D.C. 20234

Report No. 4361 Bolt Beranek and Newman Inc.

FEATURES OF THE TRANSPORT AND SESSION PROTOCOLS

John Burruss

March 1980

Prepared for:
Institute for Computer Sciences and Technology
National Bureau of Standards

Table of Contents

1 Introduction

This report holds a place in the development of communications protocols for the transport and session layers (these protocols are also known as host-to-host protocols). A description of the development sequence and of the encompassing program may be found in Federal Computer Network Protocol Standards Program: An Overview, published by the National Bureau of Standards, Institute for Computer Sciences and Technology, 1980.

It must be noted that the protocols which formed the basis for this features analysis are all in a state of evolutionary development; the descriptions found in this report may be put out of date at any time.

1.1 Feature Analysis

A feature analysis is a means of determining the kernel set of essential features of a protocol, along with the clusters of value-added features which will support various application categories. Among the criteria for selecting the kernel and ranking the value-added features are relative cost-effectiveness, level of demand, implementation expense, elegance, and sufficiency. At the end of the exercise, we will be left with the kernel set of features which must be integrated into a basic protocol, and several levels of value-added functionality which may be integrated with the protocol to form a protocol family answering the needs of disparate applications. The reasoning used to define the kernel and added functions will form the basis for the justification of choices made in the following step in protocol design, the protocol specification itself.

1.2 Layers

The first questions to be answered in a feature analysis of transport and session protocols must be: What are transport and session protocols? Why group them together in one report? Transport and Session form layers four and five of the seven layers defined in the ISO Reference Model for Open Systems Interconnection [ISO79a]. Simply put (more complete definitions will follow in the appropriate sections), the transport layer actually moves data, while the session layer co-operates with the high-level protocols to manage and structure data movement. The two layers are conveniently considered together since together they correspond to what is commonly called the host-to-host protocol. The host-to-host protocol, considered as a unity, performs two disparate functions: that of the supplier of network service, and that of the interface for high-level protocols to obtain network service. By considering transport and session protocols separately, these functions can be distinguished, and feature definition and analysis can proceed along straightforward lines.

1.3 Distinction Between Service and Protocol Features

Before embarking on discussion or cataloging of features, we should try to clarify just what a feature is. Just what sort of things a host level protocol must do is well known; different protocols do those things in different ways. For example, most host level protocols provide a connection-oriented means for moving data. But how is this connection to be established? Should all connections be full-duplex? Half-duplex? Some flow-control scheme must regulate data transfer between transport services--but methods for this vary from protocol to protocol.

The point of mentioning this diversity (of method) within unity (of function) is to point out that there are two sorts of features in a host level protocol: let us call them _service_ features and _protocol_ features. Service features are the features of the protocol visible to the immediate user of the protocol; i.e., the service features of the host/host protocol are the services it provides to the higher-level protocol-entity using it. Protocol features, however, are for the most part invisible to the immediate user: duplicate detection is an example of a protocol feature. Carl Sunshine makes a similar

distinction between specifying protocol services and specifying the operation of the protocol itself [SUNC79]. Obviously, the distinction between service and protocol features subdivides our previous separation of transport features from session features. We must consider transport protocol features and transport service features, as well as the corresponding classes of session features.

2 Transport Service

2.1 Purpose of the Transport Service

A transport service exists to provide interprocess data transfer in a network environment. Data submitted to the transport service must be delivered to a specified destination. Furthermore, it is desirable that the transport service provide communication services between processes not necessarily resident on the same network.

2.2 The Network Environment

The nature of the communications subnetwork to which the transport layer interfaces will have a considerable impact on the operation of the layer and therefore on its protocol features. This is especially true with respect to the reliability of the subnetwork. If the subnetwork provides reliable, sequenced delivery, then the transport layer need not include error-checking, sequencing, and duplicate-detection features: the burden of reliability will be carried by the subnetwork. This is true, for example, in the ARPANET IMP subnetwork [MCQJ75]. However, if the subnetwork does not provide reliable communication, the transport layer must itself assume the burden of making transport service reliable, and perform error-checking, duplicate-detection, and sequencing operations accordingly.

Examples of transport protocols which perform these operations themselves are TCP [POSJ79b] and the CYCLADES transport protocol [ZIMH75].

In this report, we will note which protocol features are affected by the nature of the subnetwork.

2.3 Transport Features

Let us examine the consequences of this statement of purpose. First, the transport service provides communication: its primary function must then be *data transfer*. As noted in the section above, if the subnetwork does not provide reliable service, the transport layer must perform *error checking* to ensure intact delivery and *duplicate-detection* and *sequencing* to ensure in-order delivery, and must provide an *acknowledgement* mechanism to exchange information about the integrity of peer transport communication. A *flow-control* mechanism is needed to keep that communication synchronized. A transport service must provide a method for transport-users to *address* their correspondents. It is expected that corresponding transport-users will be bound together in some sort of *connection* for the lifetime of their communication. Further desirable features are: provision of *transaction* and *broadcast* services as well as connection-oriented stream service; selection of a *grade of* service to be provided during communication; some means of expediting delivery of some data which may supersede previously sent information; and finally, internetwork compatibility.

It is not expected that all transport protocols surveyed in this report will have all these features. Certain features, for example data transfer, are so basic as to be universal. Others, although desirable, are rare or even nonexistent among the protocols examined in this report.

We must divide these features according to our distinction between service and protocol features; the distinguishing test in this case is whether a feature is derived from user requirements (and is therefore perceptible from above the transport/user interface) or whether it is derived from the nature of the communication environment (and therefore lies below the user interface). Some features will seem to have both service and protocol characteristics: these we will simply assign to the class that seems most natural. In this preliminary discussion of transport features, the language of the ISO Reference Model will be adopted for its generality.

2.3.1 Transport Service Features

2.3.1.1 Type of Service

The ISO Reference Model suggests three different types of service: connection-oriented (stream) service, transaction-oriented service, and broadcast service; transaction and broadcast services are left undefined.

Stream data transfer takes place between two transport-users on a transport-connection. The connection passes through three phases distinct to the user: an establishment phase, a transfer phase, and a termination phase. Communication on a connection may be full-duplex, half-duplex, or simplex (with two connections necessary to model duplex behavior).

Transaction or datagram service entails sending a data message from one transport-user to another with a single transport service request. No connection need be established, and error-checking, sequencing, and acknowledgement may be unnecessary.

Broadcast service is the sending of a single data message to all hosts (perhaps to all listening processes on all hosts or to a single process on all hosts). Again, no connection need be established.

A type of service not mentioned in the ISO Reference Model is multi-cast service. In this case, a message could be sent to all of a specified list of correspondents, and the sending transport-user would need to make only one request for service, rather than make a request for each correspondent.

Because transaction and broadcast services are not as widely understood as connection-oriented service (on account of its historical priority, if for no other reason), descriptions of network services have traditionally been couched in terms of connections and connection service. In fact, one of the shortcomings of the ISO Reference Model itself is just that orientation in its language and definitions of transport services. As we have chosen to use the language of the ISO Reference Model, the following definitions necessarily share its connection orientation.

2.3.1.2 Grade of Service

Transport users may request a grade of service during transport-connection establishment. This would define acceptable error and loss levels, desired delay, priority levels, security, and other considerations. It may be, of course, that only one level of quality is available: for example, correct in-order delivery with some average network delay. Different grades of service are significant in the case of a user of an unreliable subnetwork who will accept the unreliability of the subnetwork in return for bypassing the extra processing performed by the transport layer.

2.3.1.3 Data Transfer

The basic function of a transport service is the delivery of data. The quality of this delivery agrees with the requested grade of service.

2.3.1.4 User Interface

The details of the user/transport interface are heavily dependent on the host environment, but a few characteristics of the interface must be specified. How is the user to be prevented from swamping the transport service with data? How is the user to be notified of the success of a send or a receive? One could imagine a very primitive interface which blocked the user until his request completed, which easily leads to deadlocks unless a timeout returns control to the user. Another choice is that a request should return an immediate synchronous reply accepting or refusing the request and should later notify the user that the request has completed.

2.3.1.5 Connection Management

The transport service establishes and terminates connections between transport-users. Termination should not interfere with the usual orderly transfer of data on the connection.

2.3.1.6 Connection Abort

The transport service may also provide a peremptory abort facility in case of unrecoverable errors perceived at any protocol level.

2.3.1.7 Expedited Delivery

Some data submitted to the transport service may supersede data previously submitted. A mechanism for expedited delivery would allow transport-users to exchange such high-priority information. The ISO Reference Model leaves undefined the effect of expedited data transfer on normal data transfer.

2.3.2 Transport Protocol Features

2.3.2.1 Error Checking

A transport service transmitting on an unreliable subnetwork must insure that data accepted for delivery is delivered correctly in accordance with the requested grade of service. For a reliable grade of service, the transport service must guarantee the integrity of each data unit delivered.

In the case of a reliable subnetwork, error checking need not be performed in the transport layer.

2.3.2.2 Duplicate Detection and Sequencing

Also in accordance with the requested grade of service, the transport service must identify and destroy any duplicate data which might have been generated by the communications subnetwork or by the sending transport service during retransmission. Arriving data may not be delivered to the receiving process until it has been sorted into the order in which the sender submitted it to the transport service.

A transport service employing a reliable, sequencing subnetwork may not need to include duplicate-detection and sequencing facilities.

2.3.2.3 Retransmission

A transport service employing an unreliable subnetwork may require retransmission by the transport layer of lost or discarded data. The sender usually decides that data has been lost after spending a certain time awaiting acknowledgement. The receiver may also request retransmission of damaged data. Retransmission will not ordinarily be required in services using reliable subnetworks.

2.3.2.4 Acknowledgement

We have noted that the transport service must be prepared to retransmit data units which are lost or damaged by an unreliable subnetwork. The transport service identifies lost or damaged units by the absence of an end-to-end acknowledgement. Acknowledgement schemes are of two sorts: positive (signaling the receipt of an intact unit) and negative (requesting the retransmission of a lost or damaged unit). Other information passed end-to-end (synchronization information, pacing windows) might be piggybacked with acknowledgements, even though the other information is functionally distinct from the acknowledgement. Similarly, the acknowledgement might itself be piggybacked on other messages, especially on data messages (this is obviously possible only when both correspondents have data to send, that is, in dialogues, not in monologues).

Transport-to-transport acknowledgement may not be required in services transmitting over reliable subnetworks.

2.3.2.5 Flow Control

Two kinds of flow control regulate the movement of data in and out of the transport service: interface flow control and peer flow control. The communicating transport services regulate the flow of data between them via peer flow control. Peer flow control is usually accomplished by some window or allocation scheme, and keeps one transport service from sending more data than the other service can accept.

Flow control exists at both interfaces to the transport service. However, flow control at the transport/network interface is part of the network layer specification and beyond the scope of this report. Flow control at the user/transport interface has been discussed as a service feature.

2.3.2.6 Buffering

The ISO Reference Model does not include any discussion of buffering in the transport layer. Buffering will, however, be a concern in many implementations. Any transport layer which buffers incoming data will need to be notified when its buffers must be flushed and accumulated data passed to the user. Going in the other direction, the user must be able to tell the transport service to submit accumulated data to the network without waiting for any more data.

2.3.2.7 Transport Addressing

A transport address is the means by which the transport service identifies transport-users. A process desiring to establish a connection must be able to identify its correspondent in a way intelligible to the transport service. Typical choices for an addressing scheme are hierarchical addressing, in which local names are concatenated with global identifiers to form a unique identifier; allocated addressing, in which each system in a network is allocated certain global identifiers (these identifiers may be assigned only once or may be assigned and reassigned dynamically); and mapped addressing, in which a defined set of global identifiers is distributed among all processes and files to which network access is required (the local system maps these global names onto its local names). Hybrid schemes are also possible.

2.3.2.8 Internetworking

It is desirable for the transport service to allow communication not only between processes on different hosts belonging to the same network, but between processes on different, interconnected networks. To accomplish this, the transport service should make as few assumptions about the nature of the underlying communication medium as possible.

Communication on one network should not be constrained by any limitations of connected networks, e.g., the obtainable reliability of the transmission medium or the maximum packet size.

2.3.2.9 Accounting

If billing for network services is done on the basis of transport services (number of transport-service-data-units transmitted, for example), the transport layer would be the logical place to put accounting services.

2.3.2.10 Security

The basic security feature expected of a transport service is that it not allow unauthorized processes to manipulate connections. This might mean defining a concept of connection "ownership"; alternative meanings of "authorized" may be appropriate. Furthermore, the transport service might provide encryption service on demand. Other security precautions could be provided if they are supported by underlying layers: routing through secure nodes or on secure links, for example.

3 Transport Features Analysis

3.1 Considerations for Feature Analysis

It is natural that we should examine protocols from three conceptual viewpoints: as the absolute minimum protocol which provides the primitive functionality of a layer; as a robust and flexible protocol which provides the commonly demanded functions of a layer; and as a protocol or protocols to some degree enhanced with special, less frequently demanded or more expensive functions. If we consider the features we have described above as the components of a protocol, then the minimal protocol will be made up of the set of features defined in their minimal sense. Each feature will have an aspect corresponding to the minimal, common, and enhanced protocols just mentioned. Thus, for each protocol feature described above we will describe a _foundation definition_ of the feature, a _kernel definition_ of the feature, and one or more _value-added features_ or enhancements. The foundation definition is a minimal definition of the feature which will still allow a protocol layer to perform its primitive function. The kernel definition is one commonly agreed upon by the protocols examined for this report. Value-added features are extensions or additions which are found in some examined protocols, but which were judged too expensive or not of frequent enough demand to justify inclusion in the kernel. These value-added features do not necessarily group themselves into layers;

the clusters of additions and extensions are not hierarchically dependent on one another, as one expects layers to be, but will form at some layer a protocol family able to satisfy the requirements of particular application categories. Any or all of foundation, kernel, or value-added features may have alternate definitions depending on the network environment in which the protocol will operate.

A foundation protocol constructed from the foundation definitions of the component features might be implementable, but would surely be uselessly primitive. Our foundation is intended to have a mental existence only; it is the basis from which protocol definition proceeds.

3.2 Transport Features Analysis

It must be admitted that the analysis of transport features leaned strongly in the direction of connection-oriented stream transfer service, perhaps at the expense of transaction and broadcast services. The near universality of stream service in the protocols studied accounts for such a bias.

3.2.1 Type of Service

FOUNDATION: The protocol shall provide any single type of data transport service.
KERNEL: The protocol shall provide a connection-oriented stream data transport service.
VALUE-ADDED FEATURES: (1) The protocol may provide a transaction transfer service. (2) The protocol may provide broadcast service. (3) The protocol may provide multi-cast service.

Connection service, as has been noted above, was nearly universal among the protocols studied for this analysis. In fact, nearly all provided no other type of service.

Several protocols provided a datagram transaction service (e.g., User Datagram [POSJ79a]). In these cases, a transaction datagram was sent in a single transport message, with no error-checking, flow control, or acknowledgement expected. This sort of transaction would put the burden for such actions on the upper-level protocol, where we feel it does not belong.

One protocol provided only a reliable transaction service: the Livermore Laboratories Delta-t protocol [WATR79b]. The protocol performs error-checking, duplicate detection, and acknowledgement. The protocol maintains the state information associated with a transaction for certain periods of time after transmission or reception to allow duplicate detection and recognition of associated messages or replies. However, for stream-type communication to take place in a Delta-t environment, some rather obscure information must be passed in packet headers to permit multi-transaction messages to flow in a simulated "connection." The main advantage of the Delta-t protocol is that a transaction may be sent reliably with only two packets (the message and its reply) although with a certain amount of increased overhead based on the timer mechanism and the requirement that state information be retained for a fixed time on both sides.

There is a sequence of three TCP-4 segments which will model transaction service. This is done by beginning the three-way handshakes for connection establishment and termination in the same segment, which will also contain the transaction data. The TCP user interface could easily be extended to allow such an exchange from a single transport service request. The three-segment TCP exchange compares favorably with the two segments used in the Delta-t protocol, with one palpable advantage: TCP's retransmission facility would allow an implementation which included the transaction interface to exchange transactions with an implementation which did not include it.

There were no instances of broadcast service among the examined protocols. Indicative of the cause of its omission is a statement by McQuillan and Cerf [MCQJ78]: "Broadcasting reliably is not yet well-understood, since the usual acknowledgement schemes become unpalatable when a single broadcast would cause hundreds or thousands of acknowledgements."

Our experience with TCP and the TCP three-segment transaction leads us to believe that transaction service could be quite inexpensively integrated into a connection-oriented protocol; the example of the Delta-t protocol, however, demonstrates that addition of connection service to a transaction protocol is likely to be complex and inelegant if not costly. For this reason connection service has been included in the kernel, with the recognition that transactions are the most desirable of the additional features, and need not be costly to add.

3.2.2 Grade of Service

FOUNDATION: The protocol shall provide a reliable, sequenced service in which messages are transmitted at a single priority level.

KERNEL: The protocol shall provide a reliable, sequenced service in which messages are transmitted at a single priority level.

VALUE-ADDED FEATURES: (1) The protocol may provide multiple levels of priority for users. (2) The protocol may provide unsequenced or unreliable services at time/cost savings to the transport-user. (3) The protocol may provide low-delay/low-throughput service or high-delay/high-throughput service.

Protocols which provide connection-oriented service almost always perform error-checking and sequencing on the data transferred if necessary. An unreliable service is limited to datagram-type service in those protocols which provide it, such as the User Datagram Protocol. Provision of different grades of service is usually dependent on the communication subnetwork: a protocol on a reliable, sequencing subnetwork will not ordinarily be able to bypass the error-checking and sequencing functions in the subnetwork. The CYCLADES transport protocol allows the user to choose whether the protocol should perform error-checking and flow-control on his connection. However, the provision of reliable service in so many of the protocols examined must indicate that demand is largest for that grade of service.

3.2.3 Data Transfer

FOUNDATION: The protocol shall allow data transfer on full-duplex connections; control information must be passed on a separate logical connection. Transmitted data will be carried in octets.

KERNEL: The protocol shall allow data transfer on full-duplex connections; data and control information may flow over the same

logical connection. Transmitted data will be carried in octets.
VALUE-ADDED FEATURES: (1) The protocol may impose no byte size format on transmitted data. (2) The protocol may allow both simplex and half-duplex connections in addition to full-duplex connections.

Most of the examined protocols impose an octet format on the transmitted data; NCP [MCKA78] was the notable exception (byte size is limited to the range 1 through 255 bits, and must be constant for the life of a connection). It is true that control and data can be separated on two logical connections, and that two simplex connections can be used to provide two-way communication. However, the tendency in protocol design seems to be towards the integration of control and data on one connection (as in TCP, CYCLADES, and NCR [NCR79]) and the provision of full-duplex communication (as in TCP and CYCLADES). The issue of a separate control channel for an out-of-band signal mechanism will be addressed later.

3.2.4 User Interface

FOUNDATION: The user interface shall be synchronous, with the provision that any request will terminate after a system-defined timeout.
KERNEL: The user interface shall be asynchronous, with immediate

notification of a request's acceptance or rejection by the transport layer, and some later notification of request completion. Notification of completion is dependent on the host operating system capabilities.
VALUE-ADDED FEATURE: The timeout in a synchronous interface may be user-defined rather than system-defined.

Although the user interface is very dependent on the particular host operating system, a few things about it can be specified. First, a synchronous interface is indeed minimal, but highly liable to deadlock; a timeout on all requests should eliminate permanent deadlock. In the asynchronous interface, the transport layer would be free to accept or reject user requests, thereby providing user/transport flow control. Notification of request completion might occur through an event queue, interrupts, or some other signaling or scheduling facility provided by the operating system.

3.2.5 Connection Management

FOUNDATION: Either correspondent shall be able to initiate connection establishment and termination.
KERNEL: Connection establishment and termination procedures shall be symmetrical. Connection termination shall not interfere with the delivery of all data sent on the connection.

Some remarks have been made above about connections: they have been defined to be full-duplex. If the protocol is enhanced to allow simplex connections, even though the establishment procedures must be different for sender and receiver, it is not safe to say that establishment is always initiated by the sender. In the case of one host collecting data from other hosts, the receiver might want to initiate connection establishment. The NCR end-to-end protocol has just such a facility. It is equally clear that either correspondent must be able to initiate termination of a connection.

In the case of duplex connections, the establishment and termination sequences can be symmetrical, since there is then no distinction between sender and receiver, unless both correspondents make a conventional agreement to assume those roles. A symmetrical procedure eliminates the errors arising from colliding requests, and is on the whole simpler and more elegant.

3.2.6 Connection Abort

FOUNDATION: The protocol shall provide a means for a correspondent to request a peremptory abort of a connection. A connection will be aborted if an expected reply is not received within a system-defined timeout.

KERNEL: The protocol shall provide a means for a correspondent to request a peremptory abort of a connection. A connection will be aborted if an expected reply is not received within a system-defined timeout.
VALUE-ADDED FEATURE: The timeout for connection abort may be user-specified.

Since connection termination has been defined so as not to interfere with data transfer, some means is clearly required to end a connection without processing all the data queued on that connection. The timeout is required to detect host failure or loss of communication line.

3.2.7 Expedited Delivery

FOUNDATION: The protocol shall provide no expedited data transfer service.
KERNEL: The protocol shall provide a method for notifying transport-users of the presence of urgent data.
VALUE-ADDED FEATURE: The protocol may arrange for the delivery of urgent data to the transport-user before the delivery of pending data on the connection.

The expedited delivery service provides, among other things, a means to interrupt the remote transport-user. The exact method in which the notification of urgent data is done depends on the host operating system (e.g., one method would be via an interrupt sent by the transport service). The user is then at liberty to treat the urgent notification as he sees fit, for example, by discarding the pending messages until he reaches the urgent information. This method is used by TCP. NCP sends the interrupt through the control connection, thereby insuring that it will not be blocked by pending data. The DECnet protocol [DEC78] actually delivers some data to the transport-user at the time of the interrupt.

Garlick [GARL76] points out that a true out-of-band interrupt need not be synchronized with data and other control transmissions, and must, in fact, be independent of data flcw control. His solution to this problem is to transmit the interrupt signal over a second logical channel. This prevents the transmission of the interrupt from interfering with data transmissions, as it does in TCP, where pending data must be flushed in order to keep data flow control from blocking the delivery of the interrupt.

The designers of the Delta-t protocol contend that no expedited transfer or out-of-band interrupt is necessary when transport associations are easily set up and terminated. This is apparently true of all transaction systems.

3.2.8 Reliability Issues

FOUNDATION: The protocol shall perform error-checking, sequencing, and duplicate detection. The receiving transport layer shall acknowledge the correct receipt of all transmissions. If acknowledgement of a transmission has not been received after a system-specified timeout, the information shall be retransmitted. These functions need not be performed if they are redundant with the operation of the communications subnetwork on an end-to-end basis.

KERNEL: The protocol shall perform error-checking, sequencing, and duplicate detection. The receiving transport layer shall acknowledge the correct receipt of all transmissions. If acknowledgement of a transmission has not been received after a system-specified timeout, the information shall be retransmitted. These functions need not be performed if they are redundant with the operation of the communications subnetwork on an end-to-end basis.

VALUE-ADDED FEATURES: (1) Error-checking, sequencing, and duplicate-detection mechanisms may be bypassed if the transport-user so requests in his grade-of-service specification. (2) The retransmission timeout may be user-specified. (3) Data may be repackaged for retransmission.

These features are obviously dependent on the nature of the underlying subnetwork.

Negative acknowledgement alone is not sufficient to guarantee the integrity of a connection; positive acknowledgements will guarantee it. To allow the receiver to request retransmission of damaged data or data it thinks is lost, as in the NCR protocol, is to place the burden of reliability on the receiver. This increases traffic over the network (as the receiver must acknowledge correctly received data as well as request retransmission of damaged data).

If value-added feature (3) is included in the protocol specification, it is clear that each data byte must be identifiable, since, if only messages bear identification numbers, two messages cannot be concatenated for retransmission. Such repackaging has the advantage of reducing message leader overhead. This can be very significant, for example, in the case of TCP (which allows re-packaging for retransmission), where each message carries a leader of 288 bits.

3.2.9 Flow Control

FOUNDATION: The protocol shall provide for peer flow control.

KERNEL: The protocol shall provide for peer flow control.

VALUE-ADDED FEATURES: (1) The transport-user may bypass the protocol's peer flow control if he has so specified in his request for a certain grade of service. (2) The transport buffers allocated to a connection may be varied to provide the transport user with the requested grade of service.

All examined protocols included some method of flow control, with the exception of the User Datagram Protocol. The CYCLADES transport protocol allows the user to bypass the transport layer's flow control mechanism; however, in both the CYCLADES and the User Datagram Protocols, the user is expected to provide his own flow control. The TCP and ECMA [ECMA79a] protocols use a window mechanism to advise the sender of the receiver's capability to receive data. In both cases the window limits are included in the acknowledgement messages. A window is a range of data sequence numbers acceptable to the receiver. Generally, anything to the left of the window (i.e., out of range on the low side) is redundant, and anything to the right of the window (out of range on the high side) exceeds the receiver's resource allocation. The NCP, IBM-SNA [AHUV79], and CYCLADES protocols are based on a credit-allocation scheme; the effect is similar to the window mechanism. In the credit-allocation system, the receiver tells the sender how much data he has allocated resources for. The sender may transmit that much but no more: in essence, the "credit limit" must not be exceeded.

Because of transmission delay the window or the credit information will always be at least slightly out of date. Thus, it is reasonable to transmit the flow control information with every message transmitted in an attempt to keep that information as current as possible. Therefore, all data messages, control segments, and acknowledgements might bear the pacing parameters.

In all the protocols examined, if the allocated credit falls to zero, serious delays will be introduced. With zero credit or with a closed window, neither process would normally be transmitting, so even when the window opened or the credit raised, there would be no way of informing the correspondent. Therefore, the sender must periodically send a message giving the current pacing parameters to provoke some reply from the receiver.

The NCR end-to-end protocol contains a second means of flow control used for emergency stop: the Receiver-Not-Ready indication, which is a primitive mechanism like XON/XOFF. While this is the least expensive flow control scheme (one bit/packet), it has serious flaws which are obvious if one translates the Receiver-Not-Ready bit into a window report. The R-N-R bit off means the receive window is open; R-N-R bit on means the window is completely shut. We have mentioned above the serious effects a closed window will have on performance; the consequences of XON/XOFF flow control will similarly restrict performance.

3.2.10 Buffering

FOUNDATION: The transport layer will flush its internal buffers and deliver accumulated data to the user, or to the network, on demand.
KERNEL: The transport layer will flush its internal buffers and deliver accumulated data to the user, or to the network, on demand.

An implementation of a transport layer is not required to buffer either incoming or outgoing data. Yet it is clear from experience that buffering will commonly be required for reasons of efficiency. For example, to interrupt the transport-user upon the arrival of every incoming packet (even though other packets might be in transit and the received packet itself contain no information which could be acted upon without information from packets yet undelivered) imposes significant system overhead on the transport layer without enabling any useful processing to take place. The usual solution is to hold data until a significant amount has arrived before notifying the transport-user. A mechanism to force data delivery costs little or nothing to provide--and if it is not provided no implementation of the protocol can buffer data. Hence it seems worthwhile to define such a facility for those who require it.

TCP and the CYCLADES protocol force delivery by marking a segment of data "end-of-letter," and NCR calls its marker "end-of-message." The British Post Office Protocol [BPO79] clears buffers by transmitting the special "PUSH" packet. Comparing these two schemes, one notes that the "end of letter" marker requires but one bit in every packet whereas the PUSH packet requires transmission of (at least) the minimum-sized packet every time buffers are to be cleared.

3.2.11 Transport Addressing

FOUNDATION: There shall be a set of fixed global names with a fixed mapping onto transport-users.
KERNEL: Transport-addresses shall be the concatenation of a network-wide host address with local allocated names which are dynamically associated with transport-users.
VALUE-ADDED FEATURE: A global network name may be prefixed to the transport address to allow internetwork addressing.

The kernel definition of transport addressing, while simple enough, is not at all flexible. Consider the case of a new transport-user joining the community: all hosts must update their addressing tables and make the new information available to the users. The hierarchical concatenation of host address with a dynamically assigned local transport-address is far more flexible, but depends on the community to agree on the meanings of at least a few local names so that common resources can be accessed--this is the idea of the well-known socket familiar to NCP, TCP and DECnet.

Since a transport-user may participate in several connections, his address alone is not sufficient to identify a single connection. TCP identifies the connection by the concatenation of both transport-addresses. NCP requires a complicated Initial Connection Protocol to allocate a unique identifier for a connection. The TCP identification scheme is evidently simpler and more elegant, and it allows for agreement between transport-users even in cases where the connection establishment requests collide.

Internetwork addressability may be purchased at the cost of a few bits in the transport-address. Among the protocols examined, TCP and the User Datagram Protocol provide internetwork addressing.

3.2.12 Accounting

FOUNDATION: No accounting takes place in the transport layer.

KERNEL: No accounting takes place in the transport layer.

VALUE-ADDED FEATURE: The transport layer may keep accounts of network use.

As none of the protocols examined specified any accounting procedures and the ISO Reference Model does not locate accounting procedures within the Open Systems Environment, we felt it best to postpone a kernel definition pending further discussion of the issue. It is nevertheless clear that the transport layer is the only host-level layer which can do "per-packet" accounting.

3.2.13 Security

FOUNDATION: The protocol shall enforce no security measures.

KERNEL: The protocol shall enforce no security measures.

VALUE-ADDED FEATURES: (1) The protocol shall not allow unauthorized use of a connection. (2) The transport service may provide encryption and decryption of transmitted data on demand. (3) The protocol may support routing through secure links or nodes if such a service is available from the subnetwork. (4) The protocol may support remote verification of the receiver.

Again, as the locus of security in the Open Systems Environment is unfixed, we intend to let a kernel definition wait on further discussion. All protocols presumably contain some mechanism to prevent unauthorized use of connections, although the meaning of "unauthorized" is more or less dependent on the host operating system facilities for security, identification, and authorization. No protocol examined provided encryption or decryption services, secure routing, or remote validation.

4 Session Service

4.1 Purpose of Session Services

The purpose of the session layer is to manage data traffic between cooperating high-level protocol entities. The cooperative relationship between these entities is known as a session. The session layer forwards requests for data transfer to the transport layer.

4.2 Session Features

The first service performed by a session layer must be the administration of _sessions_. Sessions must be _established_ and _released_. Furthermore, one may expect an emergency _abort_ facility. The two cooperating session layers must be able to address each other through _session_-_service_-_access_-_points_. The session layer controls the delivery of data to the high-level user (_quarantining_), controls the interactions between high-level users (_dialogue management_), and imposes a structure on the data it transfers (_data delimiting_). Some _security_ functions relevant to the operation of distributed systems may also fall within the domain of session control.

4.2.1 The Session

A session, or session-connection, is a cooperative relationship between high-level protocol entities. The session passes through three phases: establishment, data transfer, and termination. It will be noted that these phases are closely related to the phases of the transport connection (or connections) which supports the session. In the simplest case, in which the mapping between transport connections and sessions is one-to-one, the phases of the session would be equivalent to the phases of the transport connection. However, more complex relationships are possible: a session might employ more than one transport connection (to provide an expedited or interrupt data path, for example), or several sessions employ the same transport connection consecutively or possibly simultaneously. In such cases, the phases of the session would be distinct from the phases of the transport connection.

4.2.1.1 Session Establishment

Establishment of a session will include the establishment of at least one transport-connection with an appropriate type and grade of transport service and appropriate security authentication.

Certain characteristics of a session will be demanded by the initiating session-user and negotiated between corresponding session-entities. Such characteristics would include the configuration of transport connections into a session (e.g., to provide interrupt service), the type and grade of transport service to be used, the maximum size of quarantine-units, and so on.

4.2.1.2 Session Termination

Termination of a session will dissolve the cooperative relationship between session-users in an orderly way, so that no data is lost and resources of the distributed system are released appropriately.

4.2.1.3 Session Abort

The session may be aborted with possible loss of data, for example, if a higher-level protocol-entity should perceive unrecoverable errors.

4.2.2 Addressing

A session-address obviously must have at least the scope of a transport-address. We have conceived of session addressing in terms of workstation names. A workstation name is a logical name (unique within the open system), independent of the network's physical topology, which addresses a workstation. A workstation is a process or collection of processes which carries out a particular application. It is then the function of the session layer to translate a workstation name into a transport address at the appropriate host in order to establish the proper session-connection.

4.2.3 Data Transfer

The session layer calls on the transport service to transfer data.

4.2.4 Quarantining

It may happen that one session-user desires to withhold some data from his correspondent for some length of time--for example, because none of the data being generated will be significant until a certain quantity has been produced, or until some validation procedure has been completed. The data thus held back by the session layer is said to be _quarantined_ and is known as a _quarantine-unit_. The end of a quarantine-unit is interpreted to mean that the quarantined data will be made available to the receiving session-user; thus, the quarantining facility may be used to force data delivery (that is, to flush the session buffers) as well as to hold data back from delivery. The receiving session layer will not deliver any fragment of the quarantine-unit to the session-user until all the data in that unit is available for delivery; this does not mean that the unit be delivered all at once, but only that no part will be delivered until all has arrived in the receiving session layer.

A session-user may destroy a quarantine-unit he has created if he has not yet terminated it, and it has therefore not yet been delivered to his correspondent.

4.2.5 Dialogue Management

The session layer imposes a structure on the interaction between correspondents: this structure is a dialogue. There are three possible varieties of dialogue: two-way-simultaneous, two-way-alternate, and one-way (these names should indicate the dialogue structure). The two-way-alternate case requires further consideration. One correspondent at a time will have the "turn," or permission to send. At the end of an interaction, the sender will release the turn, thus allowing the other correspondent to send. Thus there must be a method for assigning the first turn, a means for exchanging the turn, and a facility for interrupting the correspondent with the turn.

The information sent during a single turn is called an <u>interaction unit</u>. When a session-user terminates an interaction unit, he at the same time relinquishes the turn.

4.2.6 Interrupt

The basic interrupt service of a two-way-alternate dialogue is the ability to demand the turn. However, this is not of much use to session-users in a two-way-simultaneous dialogue or monologue. A more generalized interrupt facility is required for these cases. According to the negotiated configuration of the session, the interrupt facility would use either the transport layer's expedited delivery service or a separate transport connection intended for interrupts.

4.2.7 Data Delimiting

The session layer imposes a minimal level of structure on the flow of data in a session. Thus Presentation Entities may assemble meaningful units of data and have these units transported by the session layer. Of course, any imposition of structure upon a raw stream of data could be deferred by each layer to the layer above until ultimately the two highest level correspondents inherit the responsibility. In a sense, this is contrary to the principles of functional layering: the point is to find useful common functions and offer them as services at the lowest possible layer. The imposition of structure on the data stream, by segmenting it into Session-Service-Data-Units (SSDUs), is just such a service.

The SSDU can be seen as a way for session-users to segment the data stream into chunks, the boundaries between which will be maintained across the network. It is the responsibility of the local user to identify the beginning and end of an SSDU (or just the end, perhaps) for the session layer. This information would be passed across the interface as parameters in session service calls. The session layer would assume the responsibility of passing this information to its peer correspondent. The remote session layer would in turn delimit the beginning and end of the SSDU for its session-user by similarly passing parameters across the interface.

The end result of this activity is that the local session-user has a certain amount of control over how data will be transferred across the remote session user interface. The foreign user may need to post some number of _receives_ to acquire data from its serving session layer (the amount of data it gets in each _receive_ is what the ISO Reference Model calls a Session-Interface-Data-Unit). When the remote session-user does the _receive_ which includes the last octet of data in the SSDU, notification is made of the End-of-SSDU. This last _receive_ contains only data belonging to the currently "open" SSDU. More data may actually be available (belonging to the next SSDU), but it will not be passed until the next receive.

4.2.8 Security

The session layer enforces certain high-level security procedures. It may validate the security domains of its correspondents, or regulate information flow between correspondents in different security domains.

4.3 Protocol Layering and the Definition of Session Features

The ISO Reference Model has been designed based on the concept of architectural layering. Section 4 of the Model describes the principles considered during layer definition. These include:

> 3) create separate layers to handle functions which are manifestly different in the process performed or the technology involved,
>
> 4) collect similar functions into the same layer,
>
> 9) enable changes of functions or protocols within a layer without affecting the other layers.

These three principles may be generalized into a principle of Layer Independence or Transparency:

The performance of a layer should not depend on information which does not affect its functions.

Attention to this principle of transparency during layer definition will result in truly independent layers, any of which may be modified or replaced without affecting any other layer. For instance, one example of proper application of the principle is found in the description of the network layer, section 5.5.2.1:

> The basic service of the network layer is to provide the transparent transfer of all data submitted by the transport layer. This service allows the structure and detailed content of submitted data to be determined exclusively by layers above the network layer.

If we apply the principle of transparency to peer interactions between correspondent protocol-entities, we will find that it forbids the division of a single function between two layers. That is, one must not require one layer to delimit data on behalf of another layer for which the resulting units have significance. Rather, the same layer for which a data unit has significance should delimit that unit. This points up two layering flaws in the definition of the session layer: the commitment and recovery unit delimiting, and quarantining. In both cases, the session layer could end up marking data for the use of the presentation layer.

Let us address the quarantining function first. The ISO Reference Model allows either the session layer or the presentation layer to perform quarantining. If the presentation layer performs quarantining, the session layer is still required to mark the quarantine-units and pass these delimiters on to the presentation layer. Clearly, this is an unacceptable division of a single function between two layers. The layers can be made independent by concentrating the quarantining function in a single layer; it is our opinion that quarantining is a function appropriate to session control, different in kind from the functions associated with the presentation layer. For that reason we will define quarantining as a session feature (we will return presently to the question of quarantine-unit delimiting).

The recovery- and commitment-units have meaning only to presentation-entities or application-entities (although the ISO Reference Model says these units are "defined and managed by presentation-entities [5.3.2.7.4]," commitment and recovery control are listed as application management functions within the application layer [5.1.3.4]). If the session layer is to transparently transmit the data it receives, it should not introduce any delimiters which must be passed to higher layers. Recovery- and commitment-units should be delimited in the same layer at which the peer-to-peer integrity protocol functions. We will define the session layer without recovery- or commitment-unit delimiting, leaving that function to the higher layer which carries out the integrity protocol.

4.4 Further Notes on Data Delimiting

Even after having made the session layer independent by concentrating quarantining in the session layer and the integrity protocol in a higher layer, the locus of quarantining is still not entirely clear: buffering of the quarantined data may take place in the local session layer, in the remote session layer, or in both. Wherever quarantining takes place, the idea of a quarantine-unit is not really useful: a quarantine-unit has no significance as a unit, and quarantining is more usefully thought of as a temporal concept. An end-of-quarantine-unit delimiter may be more naturally thought of as a token of permission to deliver the quarantined data instead of a data delimiter. In any case, the session layer must not notify the presentation layer of the end-of-quarantine-unit; to do so would violate the principle of transparency by forcing on the presentation layer information which should have no effect on its functions.

The other data unit delimited by the session layer is the interaction-unit. In this case, an end-to-end interaction-unit-delimiter is required; but the concept of an interaction is, like quarantining, temporal, and rather than considering delimited interaction-units, it may be more natural to think of the unit delimiter as an end-to-end turn-exchange token. That is, we will not speak of interaction-units, but model the exchange of turns by the exchange of the token: only the correspondent holding the token will be allowed to send. Such an idea makes it easier to consider interactions in a round-robin multiple-correspondent situation, where the unit concept is unnatural and confusing.

4.5 Session Control and Transport Transaction Service

The concept of Session Control, as presented in the current version of the ISO Reference Model, is strongly based on the use of the transport layer's connection-oriented service (which is itself the only type of transport service discussed). The definition of a session makes explicit this dependence (section 5.3.1.2.2):

> For the transfer of data between two presentation-entities, the session is mapped onto and uses a transport-connection.

In fact, the authors of the current draft seem uneasy about a session's strong identification with a transport-connection. One of several "urgent issues" is (5.4.5.1):

> What is the precise difference between 'sessions' and 'transport-connections'?

The whole notion of a session which binds two presentation-entities into a relationship of some temporal duration is an upwards extension of the idea of connections at the transport layer or virtual circuits at the network layer.

It is hard to see how two corresponding session layers could establish, manipulate, and release a session during a single transport transaction (that is, a "single-access" data transfer request to the transport service). If the session layer must make more than one transport transaction during the life of a transaction/session, it might as well open a transport-connection and go through normal session-establishment procedures rather than trying to use the other type of service. It is equally hard to see why one would establish a session for a series of transport transactions, or even how such a session would be established. How then is the session layer to use the transport transaction service?

The point of transport transaction service is to provide a means of sending a certain amount of data with a single transport access (no assumption is made about the number of network layer accesses corresponding to a single transaction). We will assume that there is an analogous single-access "transaction" primitive in the session layer, which will cause the usual address transformation, security check (if necessary), *et cetera*, to take place, but which performs data transfer of a single Session Service Data Unit in a single transport access. The receiving session layer would recognize the SSDU as a transaction requiring little or no session action and pass the SSDU on to the appropriate session-user. This idea perhaps sidesteps the issue of how the session layer uses transaction·service, by essentially promulgating transaction service transparently through the session layer. It does seem, however, to follow in near perfect analogy: the session transaction stands in the same relationship to the transport transaction as the session-connection to the transport-connection.

5 Session Feature Analysis

We must again note that the session layer protocols discussed in this part of the report are in a state of evolutionary change. This is furthermore true of the definition of the session layer itself and services it requires and provides. Descriptions found in this part of the report describe protocols as they existed during the study it describes; these protocols may change at any time.

5.1 Considerations for the Feature Analysis

In our feature analysis of the transport layer, we had a good deal of material from which to extract features and to use as fixed points for comparison: this material included transport protocols from commercial vendors, from government networks, and contributions from national and international standardization groups. As a result we felt that our analysis had not wandered far from the thinking of the networking community. When we turned to the session features analysis, however, we found a far smaller pool of material with which to start. This scarcity is understandable since session control has until recently been a relatively undefined area in the ISO Architecture.

Of the commercial network architectures available to this study, only NCR [NCR79] has a distributed layer which includes any session functions (although this layer 5-C is more of an interface to the transport layer than a real session layer). IBM's SNA defines the concept of session and half-session, but these turn out to have more to do with layer 4 functions than with layer 5; SNA does not seem to support a distributed session layer (in the sense of the ISO Reference Model) [CORF79] and [MCFJ76]. The only fully-defined session layer (available to this study) which exploits all the functionality of the ISO Reference Model definition is a specification by the German Hahn-Meitner-Institut (HMI) [VOGF79]. The European Computer Manufacturers Association has distributed two working papers giving provisional sketches of the services offered by a session layer, [DEBC79] and [ECMA79b]. Beyond these few attempts at real specification of a session layer, we have had to rely on the ISO Reference Model itself, and explications of the Model by Bachman and Canepa [BACC79] and desJardins and White [DESR78].

Thus it is easy to see that features analysis cannot proceed along the comparative lines of the transport layer analysis. Rather than begin by grouping already defined features into clusters based on our judgement of relative cost and usefulness, we ourselves shall have to define groups of features and try to discuss them in terms of costs and benefits. Benefits will include such considerations as the power of a given feature and proposed uses for it. Costs will include complexity of protocol machinery required to provide a feature and the degree of effective use of transport services. We shall follow a format for discussion the reverse of that which we used in the transport analysis: through our discussion we shall arrive at definitions for the minimum foundation and standard features, which follow at the end of each section.

5.2 A Minimal Foundation for the Session Layer

The minimal foundation for any protocol layer will be the minimal functionality which must be included in the layer for a host to participate in distributed processing. This is important with regard to the session layer, since many network architectures developed before the ISO Reference Model define no session layer. Regarding such architectures, we must ask if session control functions are being carried out somewhere else in the architecture? If so, where? How can such functions be concentrated in an identifiable layer? These questions are of particular interest with respect to the government networking community, especially ARPANET and TCP-4 developers. It has been the practice of such systems to concentrate transport functions in a host-to-host protocol and to place presentation functions in the end-user application processes. To what extent can we say a session layer exists in such an architecture?

The minimum functionality for a session layer is merely the transparent promulgation of transport services. That is, there would be no difference between a session and a transport-connection; a session establish request and a transport-connection open would be the same, for example. A clever implementation of such a session layer might take advantage of this one-to-one relationship between layer services and implement only the transport service requests. Thus the session layer would be empty or null, since the implementation contains no code and transport and session services are apparently identical. From an architectural viewpoint, then, one can speak of a session layer existing in such a case, but to do so is not particularly meaningful.

In short, we need not address the question of a minimum foundation for the session layer. Experience with network architectures which do not define a session layer shows that the minimum session requirement for participation in distributed processing is an empty layer.

5.3 Session Establishment

Several characteristics of a session may not be determined until session establishment. Many of these characteristics will be services requested by the session-user, such as a particular grade of transport service, the first turn in a dialogue, or similar requests. However, it is not sufficient for one correspondent to demand a session of a certain type; that type must be acceptable to both session-users, and agreed upon by both session layers. In order to allow the session layers to establish an acceptable session, the possible options must be encoded and a negotiation procedure must be defined in our protocol. This procedure must meet several requirements: neither correspondent should be forced to accept a session which does not meet its requirements, the negotiation should not result in a non-terminating series of requests and responses, and the negotiation procedure should not introduce a master/servant distinction between the session layers.

Neither the NCR Layer 5-C nor the HMI session protocol defines a negotiation procedure or service options. The ARPANET protocols TELNET [DAVJ77] and THP [POSJ76], which perform some session-like functions, do allow for negotiation of service options.

We can imagine two different styles of negotiation: a request/response negotiation dialogue (as used in TELNET and THP) or a rule-directed scheme in which the attributes of the session are determined by applying a fixed set of rules to the session requirements of both session layers.

The request/response procedure has the advantage of being well-known and widely used in TELNET's and THP's WILL/WONT, DO/DONT scheme; however, it may require the introduction of a priority distinction between correspondents or result in a non-terminating sequence of exchanges. Such situations might arise, for example, if both session layers simultaneously demanded the first turn in a dialogue. Furthermore, the negotiation of any single option cannot happen faster than the round-trip time of the request and response (this for an uncontested option requiring only a single exchange). The flurry of exchanges one would expect when setting up a session could add appreciably to establishment delay and the transmission overhead of establishment.

The rule-directed procedure would require only two exchanges: one of attribute codes, another accepting or rejecting the session. However, since both session layers must apply the same rules independently to the same input, there is some redundant computation. It is clear that some care must be devoted to designing the rules.

We do not propose here to make a choice between the two possibilities. That will be left to the service specification.

KERNEL: The establishment of a session will allow negotiation of various service options between session layers.

5.4 Session Termination

Release of a session is seen as a cooperative enterprise which preserves the integrity of any data still in the connection. This means that both sides must agree when to close the session. In a two-way-alternate dialogue, only the side with the turn may initiate release procedures. In a two-way-simultaneous dialogue or a monologue, either side should be able to initiate release.

Both NCR Layer 5-C and the HMI protocol behave in this manner.

KERNEL: Either one of a pair of cooperating session-users must be able to initiate release of the session, with the exception of the correspondent without the turn in a two-way-alternate dialogue. Session release will not cause loss of session data.

5.5 Session Abort

A session-user should also be able to unilaterally abort a session--but such a termination may cause loss of data still in transit. One use of this facility would be to end a session if the higher-level protocol entities discover unrecoverable errors.

Neither NCR nor the HMI protocol have a user abort service.

KERNEL: Either session-user may unilaterally abort a session; this termination will not respect pending data in the connection.

5.6 Addressing

It is common, on ARPANET systems, in DECnet, and in systems using TCP-4 as a transport layer, for certain widely used network services (or workstations, in the language of Open Systems Interconnection) to await connections at the same address in different hosts. For example, an FTP server might be contacted through port 10 at Host A, port 10 at Host B, and port 10 at Host C as well. Any customer seeking FTP service is familiar with this well-known address, called a well-known socket. A simple type of session addressing would be to translate a string workstation name into a logical transport address using tables of well-known sockets. For example, a user desiring to use FTP from or to a host called BBN-UNIX * might specify "FTP@BBN-UNIX." If networks have been interconnected, it might be necessary to specify "ARPANET:FTP@BBN-UNIX." The many conceivable variations in format of the string are not significant.

* UNIX is a trademark of Bell Laboratories.

Session address translation is a local service provided to a session-user and has no peer-to-peer significance.

KERNEL: The session layer will translate symbolic names into the proper transport-address.

5.7 Data Transfer

When a user requests the session layer to transfer data, he may have to include other information not related to the actual transfer of the data--information concerning data delimiting or dialogue turn exchange, for example. Such extra information, while an essential part of session service, is distinct from data transfer _per se_. The session layer will transparently forward the data to the transport layer, which will actually transfer it. Of course, if necessary, the session layer may indicate control conditions in the data stream as it is submitted to the transport service.

KERNEL: The session layer will transparently forward to the transport layer data to be transferred.

5.8 Quarantining

Some peer-to-peer mechanism is necessary for proper quarantining. We have defined a quarantine-unit as a portion of data none of which will be delivered until all of it is available. It does not seem reasonable to require that all data lie in a quarantine-unit; one should be able to turn quarantining on and off as needed. If quarantine-units are not contiguous, the data will have to be marked at the beginning of a quarantine-unit and at the end of the unit.

The ISO Reference Model leaves open the question of where quarantining takes place; that is, which session layer holds the data until the sending session-user closes the unit. Even if the sending session layer accumulates a whole quarantine-unit before beginning transmission to the receiver, the receiving session layer must have a buffer big enough to hold the whole unit, since the session layer must have the last byte of the unit in hand before it can deliver the first byte. The sender must have equally large buffers for holding the accumulating unit. Nevertheless there are some benefits to having the sender perform quarantining on outgoing data. First, it evenly distributes buffer use between two session layers: each will quarantine its own outgoing data. Second, no protocol machinery is required for cancelling a quarantine-unit--this becomes a local procedure. Finally, holding an open unit locally evades the problem of the relationship of interrupts to open quarantine-units. To prevent deadlock, the sender must not transmit a quarantine-unit larger than the receiver is prepared to accept. Therefore each session layer must negotiate the maximum size of quarantine-units it will transmit.

Neither the HMI protocol nor NCR layer 5-C define quarantining procedures. The ECMA working paper on session control defines the existence of quarantine-units without mentioning any quarantining procedures or localizing the action in either of the communicating hosts.

KERNEL: Quarantine-units may appear intermittently in the data stream. The beginning and end of a quarantine-unit must be marked. None of a quarantine-unit may be delivered to the receiving session-user until the receiving session layer is prepared to deliver the last octet of the quarantined data. None of a quarantine-unit may be transmitted by the sending session layer until that layer is prepared to transmit the last octet of the quarantined data. Cancellation of the currently open quarantine-unit is performed locally. The size of receive quarantine-unit buffers will be negotiated during session establishment, and no quarantine-unit larger than a receiver's buffer shall be sent.

5.9 Dialogue Management

We have already suggested that the type of dialogue to be used in a session and the initial turn (if the dialogue is two-way-alternate) should be negotiated during session establishment. No management is necessary in two-way-simultaneous dialogues, and during a monologue, the local session layer need only refuse send or receive requests appropriately as the local user is the sender or receiver of the monologue.

In two-way-alternate dialogues, however, some management procedures are required. A session layer will indicate when its session-user relinquishes the turn by a control indication (end-of-turn). Presumably, a session layer will notify its user that he is now permitted to send when it receives the end-of-turn indication. Each session layer will refuse send or receive requests appropriately as the user does or does not have the turn.

During a two-way-alternate dialogue we have assumed the possibility of one user interrupting the other. This is done by transmitting a "demand turn" message--either on the interrupt channel or by transport expedited service. Some rule must be established for what action must be taken on receipt of the demand, lest the structure of the dialogue break down.

Neither NCR layer 5-C nor the HMI protocol defines dialogue management procedures.

KERNEL: The session layer shall enforce constraints on its session-users appropriate to the nature of the dialogue in progress and the possession of the turn. The turn shall be exchanged by means of an end-of-turn indication. The demand-turn request shall not be ignored, but the session layer with the turn must immediately relinquish it. The end of a turn will automatically close the open session-service-data-unit or quarantine unit.

5.10 Interrupts

We have noted the use of interrupts with respect to the demand-turn facility. This, however, is only of use during two-way-alternate dialogues, and then restricted to a narrow meaning. A general interrupt facility would be of use in all types of dialogue. We could imagine provision of a signal which would deliver no data but provide only an interrupt indication. However, both transport expedited service and a separate interrupt channel allow sending more than a simple signal--an interrupt could carry with it some amount of data. The interrupt facility could use whatever type of expedited service had been arranged when the session was established.

The HMI protocol allows session-users direct access to transport expedited service, which gives it approximately the power of the interrupt with data described above. However, that protocol also proposes a purge facility, noting that purge may make expedited transfer superfluous. We felt that the interrupt is more general and, in many applications, more useful than a purge service. The action to be taken on receipt of an interrupt (which perhaps might be to effectively purge the connection) would be left up to the correspondents and the content of the data sent with the interrupt, if any.

KERNEL: The session layer shall, on request, transmit an interrupt message which may contain a certain amount of data, using the expedited transfer service or interrupt channel agreed upon during session establishment.

5.11 Data Delimiting

The only data delimiting facility of the HMI protocol is inherent in its data transfer request: the session-user sends or receives a single Session-Service-Data-Unit in what it calls a <u>NSSDU</u> request. This means that the SSDU and the Session-Interface-Data-Unit are identical. This identity is not essential to the operation of the layer, however, if one considers the SSDU to be a low-level data structuring facility. There is no reason a SSDU should not cross the session/user interface in several pieces. We require only that each SSDU be distinguishable to the session-user.

Our session protocol will mark the end of each SSDU. Since, in agreement with the ISO Reference Model, SSDUs will be continuous through the data stream, there will be no need to mark the beginning of an SSDU.

Finally, the hierarchy of data delimiting functions must be defined. Clearly, relinquishing the turn requires delivery of any backed-up or quarantined data, so end-of-turn entails end-of-quarantine-unit (if there is an open quarantine-unit). As it is supposed that quarantining affects only an integral number of SSDUs, end-of-quarantine-unit entails end-of-SSDU as well.

KERNEL: The session layer shall divide the data stream into a sequence of SSDUs according to the directions of the sending session-user. Incoming SSDUs shall be distinguishable to the receiving session-user. No interpretation shall be placed on the SSDU by the session layer.

5.12 Security

None of the session protocols available to this study defines any session-level security procedures. We shall leave this area undefined until further study clarifies the locus and requirements of security in the Open System environment.

KERNEL: No security procedures are defined in the session layer.

5.13 Session Transaction Service

As we noted earlier, no session protocol available to this study has addressed the question of how a session layer uses transport transaction service. Most aspects of session control--quarantining, dialogue management, and so forth--make sense in the context of a data stream; it makes little sense to apply such control functions to single-access transactions. We have argued that the session-user should essentially be given direct access to the transport layer's transaction requests.

When a session-user requests that a transaction be sent, the session layer might mark data to indicate that session establishment procedures are unnecessary. The data in the transaction will merely be handed to the receiving session-user: no long-term association is created.

We consider the transaction to be a single Session-Service-Data-Unit because, although data delimiting is primarily a stream-oriented function, it is convenient to imagine <u>all</u> data divided into SSDUs. Although any option which might apply to normal data transmission might be legal in a transaction, not all of them would be meaningful in that context (for example, an end-of-turn indication).

Transactions might be delivered to session-users which have declared a desire to receive them. It seems that applications would normally deal exclusively in either stream or transaction service, but not mix them.

KERNEL: The session layer will deliver single-access transactions on behalf of its users by means of the transport layer's transaction service.

References

[AHUV79] Ahuja, V. "Routing and Flow Control in Systems Network Architecture." _IBM Systems Journal_, 18 (1979), 298.

[BACC79] Bachman, Charles, and Mike Canepa. "The Session Control Layer of an Open System Interconnection." OSIC/TG 6/79-10.

[BPO79] (British) Post Office Packet Switching Study Group 3. _A Transport Service_. BIG/CP(79)7, 4 April 1979.

[CBEMA79] Computer and Business Equipment Manufacturers Association, Task Group X3S33 on Data Communications Formats. _Third Draft Proposed American National Standard for Heading Format Structure for Code Independent Communication Headings_. 281F/3, OSIC/80-10, November 29, 1979.

[CORF79] Corr, F.P., and D.H. Neal. "SNA and Emerging International Standards." _IBM Systems Journal_, 18 (1979), 244.

[CRAR79] Crawford, Robert. "Comparison of Transport Services." SPARC/OSIC/TG6/79-18, December 20, 1979.

[DAVJ77] Davidson, J., W. Hathaway, J. Postel, N. Mimno, R. Thomas, and D. Walden. "The ARPANET TELNET Protocol: Its Purpose, Principles, Implementation, and Impact on Host Operating System Design." Fifth Data Communications Symposium, 1977. Reprinted in [MCQJ78].

[DEBC79] de Bourbon, [Charles]. "Session Control Overview." European Computer Manufacturers Association, ECMA/TC23/79/57, August 1979.

[DEC78] _DECnet/Digital Network Architecture_: _Network Services Protocol_ (_NSP_). Functional Specification, Version 3.1. Digital Equipment Corp., March 1978.

[DESR78] desJardins, Richard, and George White. "ANSI Reference Model for Distributed Systems." IEEE Compcon 78, 5-8 September 1978.

[ECMA79a] European Computer Manufacturers Association. _Standard ECMA Transport Protocol_, First Draft. ECMA/TC24/79/70, ECMA/TC23/79/46, July 1979.

[ECMA79b] European Computer Manufacturers Association, TC 23 (ad hoc group on Common Services). "Working Paper on Session Control." ECMA/TC23/79/70, OSIC/TG 6/79-7, September 1979.

[FLEJ78] Fletcher, John G., and Richard W. Watson. "Mechanisms for a Reliable Timer-Based Protocol." Computer Networks, 2 (1978), 271-290.

[GARL76] Garlick, Lawrence L. "Out of Band Control Signals in a Host-to-Host Protocol." Network Information Center, September 1976. Request for Comments 721. Network Information Center 36636.

[GARL77] Garlick, Lawrence L., Raphael Rom, and Jonathan B. Postel. "Reliable Host-to-Host Protocols: Problems and Techniques." Proc. Fifth Data Communications Symposium, September 1977.

[ISO79a] International Standards Organization. Reference Model of Open Systems Interconnection. ISO/TC97/SC16 N227, August 1979.

[ISO79b] International Standards Organization. Transport Service Functions and Services. ISO/TC97/SC6 N1861, September 1979.

[MCFJ76] McFadyen, J.H. "Systems Network Architecture: An Overview." IBM Systems Journal, 15 (1976), 4.

[MCKA78] McKenzie, Alex. "Host-to-Host Protocol for the ARPANET," ARPANET Protocol Handbook. Ed. Elizabeth Feinler and Jonathan Postel. Defense Communications Agency, January 1978. Network Information Center 7104.

[MCQJ75] McQuillan, John M. "The Evolution of Message Processing Techniques in the ARPA Network." 1975. Reprinted in [MCQJ78].

[MCQJ78] McQuillan, John M., and Vinton G. Cerf. Tutorial: A Practical View of Computer Communications Protocols. IEEE Computer Society, 1978. IEEE Catalog No. EHO 137-0.

[NCR79] NCR Corporation. Personal Communication, 1979.

[POSJ76] Postel, Jonathan B., Larry L. Garlick, and Raphael Rom. *Terminal-to-Host Protocol Specification*. Augmentation Research Center, Stanford Research Institute, July 15,1976.

[POSJ79a] Postel, Jonathan B. *User Datagram Protocol*. University of Southern California, Information Sciences Institute, 2 May 1979. Internet Experimental Note 88.

[POSJ79b] Postel, Jonathan B. *Transmission Control Protocol*. University of Southern California, Information Sciences Institute, August 1979. Internet Experimental Note 112.

[POSJ79c] Postel, Jonathan B. *Internet Protocol*. University of Southern California, Information Sciences Institute, August 1979. Internet Experimental Note 111.

[SLOL79] Sloan, Lansing J. "Limiting the Lifetime of Packets in Computer Networks." Lawrence Livermore Laboratory, Preprint UCRL-82825 Rev. 1, November 27, 1979.

[SUNC79] Sunshine, Carl A. *Formal Methods for Communication Protocol Specification and Verification*. The Rand Corporation, September 1979. No. WD-335-ARPA/NBS. (Working Draft)

[VOGF79] Vogt, F., E. Dregger, H. Eckert, and B. Lausch. *Specification of a Transport and Session Layer Protocol Based on the Message Link Protocol, Version 1.0*. Hahn-Meitner-Institut fuer Kernforschung G.m.b.H., Bereich Datenverarbeitung und Elektronik. Berlin September, 1979.

[WATR79a] Watson, Richard W., and John G. Fletcher. "An Architecture for Support of Network Operating System Services." The Fourth Berkeley Conference on Distributed Data Management and Computer Networks, August 28-30, 1979. Lawrence Livermore Laboratory, Preprint UCRL-82568 Rev. 1, August 8, 1979.

[WATR79b] Watson, Richard W. *Delta-t Protocol Preliminary Specification*. Lawrence Livermore Laboratory, November 1979.

[WATR80] Watson, Richard W. "Rough Comparison of TCP and Delta-t Connection Management." Private Communication, January 4, 1980.

[ZIMH75] Zimmerman, Hubert. "The CYCLADES End to End Protocol." IEEE Fourth Data Communications Symposium, October 1975.